7-2021

3D Object Detection Via 2D LiDAR Corrected Pseudo LiDAR Point Clouds

Saurabh Mahendra Sonje
ss7876@rit.edu

3D Object Detection Via 2D LiDAR Corrected Pseudo LiDAR Point Clouds

Saurabh Mahendra Sonje

3D Object Detection Via 2D LiDAR Corrected Pseudo LiDAR Point Clouds

Saurabh Mahendra Sonje

July 2021

A Thesis Submitted
in Partial Fulfillment
of the Requirements for the Degree of
Master of Science
in
Computer Engineering

RIT Kate Gleason College of Engineering

Department of Computer Engineering

3D Object Detection Via 2D LiDAR Corrected Pseudo LiDAR Point Clouds

Saurabh Mahendra Sonje

Committee Approval:

Prof Dr. Guoyu Lu | *Advisor* Date
Chester F. Carlson, Center for Imaging Science

Prof Dr. Andres Kwasinski Date
Department of Computer Engineering

Prof Dr. Alexander Loui Date
Department of Computer Engineering

Acknowledgment

I would like to thank my advisor Prof. Dr. Guoyu Lu for his constant support and guidance throughout my academic and research work. I am thankful to Dr. Andres Kwasinski and Dr. Alexander Loui for being on my thesis committee. I am forever grateful to my family and friends; their love and support has always motivated me to do more.

To Aai, Baba, Niraj and Manasi... Thank you for always being there. Without you, this wouldn't have been possible.

Abstract

The age of automation has led to significant research in the field of Machine Learning and Computer Vision. Computer Vision tasks fundamentally rely on information from digital images, videos, texts and sensors to build intelligent systems. In recent times, deep neural networks combined with computer vision algorithms have been successful in developing 2D object detection methods with a potential to be applied in real-time systems. However, performing fast and accurate 3D object detection is still a challenging problem. The automotive industry is shifting gears towards building electric vehicles, connected cars, sustainable vehicles and is expected to have a high growth potential in the coming years. 3D object detection is a critical task for autonomous driving vehicles and robots as it helps moving objects in the scene to effectively plan their motion around other objects. 3D object detection tasks leverage image data from camera and/or 3D point clouds obtained from expensive 3D LiDAR sensors to achieve high detection accuracy. The 3D LiDAR sensor provides accurate depth information that is required to estimate the third dimension of the objects in the scene. Typically, a 64 beam LiDAR sensor mounted on a self-driving car cost around $75000. In this thesis, we propose a cost-effective approach for 3D object detection using a low-cost 2D LiDAR sensor. We collectively use the single beam point cloud data from 2D LiDAR for depth correction in pseudo-LiDAR. The proposed methods are tested on the KITTI 3D object detection dataset.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Convolutional Neural Networks have proven to be outstanding at extracting features from images, videos and texts. These features have been utilized to perform crucial vision tasks such as object detection, image classification, segmentation, depth estimation, and more. These tasks have found applications in domains such as self-driving cars, robotics, gaming and healthcare. The performance of supervised Convolutional Neural Network (CNN) based methods is largely dependent on the quality of labelled data available for training the neural network. In case of 3D object detection, additional accurate depth information is required to estimate the 3D coordinate of the objects in the scene. The 'z' coordinate or the third dimension is necessary for the moving objects in the scene to effectively plan their motion around other objects.

According to sources, the global autonomous car market has observed a Compound Annual Growth Rate (CAGR) of 12.7% since 2015 and is expected to gain momentum by 2028. An autonomous driving vehicle on different levels of autonomy relies on information from a number of sensors that are mounted on or inside the vehicle that assist in parking the vehicle, blind spot detection, lane warnings, cruise control, object recognition, decision making, and more. Sensors such as camera, RADAR (Radio Detection and Ranging), Ultrasonics, LiDAR (Light Detection and Ranging) are commonly used for environment perception. In context

of autonomous driving, multiple beam 3D LiDAR sensors are very popular and are used to obtain high-resolution 3D map of the environment that provides accurate depth information. Such sensors ensure high accuracy that is needed for reliable and safe driving. LiDAR sensors are robust in nature, independent of environmental factors and measure data in high resolution. Camera based approaches for depth estimation use either monocular or stereo images but are not as highly accurate as 3D LiDAR. The accuracy of estimated depth values using camera-based approaches decreases as the distance between the viewpoint and the object in the scene increases. It is difficult to estimate depth from 2D images using only local image features as accurate depth estimation requires us to view the image in a global context. On the other hand, LiDAR sensors do a great job to provide accurate depth data but they come with a tradeoff of high cost. The higher the number of beams, the more is the cost of the LiDAR sensor. Typically, a 64 beam LiDAR sensor mounted on a self-driving car costs around $75000. That is about two times the average cost of a car in the United States of America.

To make self-driving cars more affordable in the future, it is necessary to curb the high cost of 3D LiDAR sensors. As an alternative to the LiDAR sensor, Wang et al.,[11] proposed the concept of “pseudo-LiDAR”. A pseudo-LiDAR point cloud is similar to a LiDAR point cloud but does not actually require a LiDAR sensor for its generation. 3D object detection using pseudo-LiDAR method has shown potential improvement in the detection accuracy for image-based methods.

1.2 Motivation

3D object detection and localization are one of the most critical tasks of autonomous vehicles. For a safe travel experience, it is essential for the vehicle to detect objects such as cars, pedestrians and cyclists accurately while in motion. Accuracy of 3D object detection is highly dependent on the depth information of objects in the scene. Mostly, the state-of-the-art object detection methods [23,24,29] use LiDAR as the source for accurate depth information. The invent of pseudo-LiDAR [11] has proven to be a game-changer for image-based methods as it has led to an increase in the detection accuracy by almost 160%. Although, it is observed that a large percent of the high detection accuracy is for objects, specifically cars, that are in a range of 30m from the viewpoint. The objects that are farther i.e., greater than 30m from viewpoint, have a low detection accuracy for an Intersection Over Union (IoU) value of 0.7. It is observed that there lies some disparity in the depth values of points for far-away objects in pseudo-LiDAR. To correct the disparity, Wang et al., in pseudo-LiDAR++ [12], introduced a depth correction algorithm based on K-Nearest Neighbors and KD-Tree that uses depth information from 4 beam LiDAR as a ground truth, to iteratively correct the depth values of the whole point cloud by correcting the corresponding depth values of points in pseudo-LiDAR.

In a similar manner, a low-cost 2D LiDAR can be leveraged to perform depth correction of pseudo-LiDAR. 3D object detection using 2D LiDAR is a potential area of research that is yet to be fully explored. 2D LiDAR sensors work in the same manner as 3D. They emit single laser beam that scans the surrounding and gathers depth information along the X and Y axis. The depth information from 2D LiDAR can be used to generate a 3D point cloud by employing the camera calibration matrices or the camera intrinsic and extrinsic. As an alternative to 3D LiDAR, the power of 2D LiDAR and pseudo-LiDAR can be used together to improve the existing 3D object detection methods.

1.3 Contributions

The principal contributions of this thesis research are outlined as:

- We demonstrate a cost-effective approach for 3D object detection using 2D LiDAR
- We introduce a novel approach for depth correction in Pseudo-LiDAR using Accelerated Coherent Point Drift algorithm.
- We modify the CenterNet 3D object detection model to train on LiDAR Birds Eye View (BEV) images
- We provide a 2D LiDAR-based approach that requires less memory and saves about 75% of the cost when compared with the Velodyne 16 beam LiDAR system

1.4 Document Structure

Chapter 2 discusses the background on Convolutional Neural Networks, 3D object detection from RGB Images, 3D object detection from LiDAR, Working principle of LiDAR, Depth Estimation and concept of pseudo-LiDAR. Chapter 3 will discuss the methodology, that includes the Network Overview, Single Beam 2D LiDAR, Depth correction method and Deep Learning models used for experimentation. Chapter 4 will outline the training dataset used along with the implementation details such as hyper-parameters, losses and evaluation metrics. Chapter 5 will analyze the qualitative and quantitative results for the proposed methods on the modified CenterNet network. Chapter 6 will include the conclusion and will give directions for future work.

Chapter 2

Background

2.1 Neural Networks

Neural Networks, also popularly known as Artificial Neural Networks (ANN), CNN or Recurrent Neural Networks (RNN), are inspired by the human brain. Similar to the human brain, a neural network comprises of many neuron-like processing units having several interconnections between them. Also, like the human brain, a neural network functions parallelly and follows a distributed process. In recent times, neural networks such as CNNs have shown great potential for application in various downstream tasks. They mainly comprise of convolution layers, pooling layers, activation layers and fully connected layers. A CNN extracts feature maps by applying filters to input images and the previous layers of output feature maps. The convolutional layers are responsible to generate feature or activation maps and the process of convolution is repeated several times to line up the features with every possible image patch. The pooling layers are used to down-sample the feature maps while still preserving the important features. This technique helps in reducing the computational load of the network. An activation function is a very critical aspect in the design of a neural network. It decides whether or not a neuron should be activated based on the weighted sum of the inputs and biases. Fully connected layers are the final layers in a neural network where each neuron is connected to every other neuron in the previous layer. They compile the features extracted from the previous layers to predict the final output.

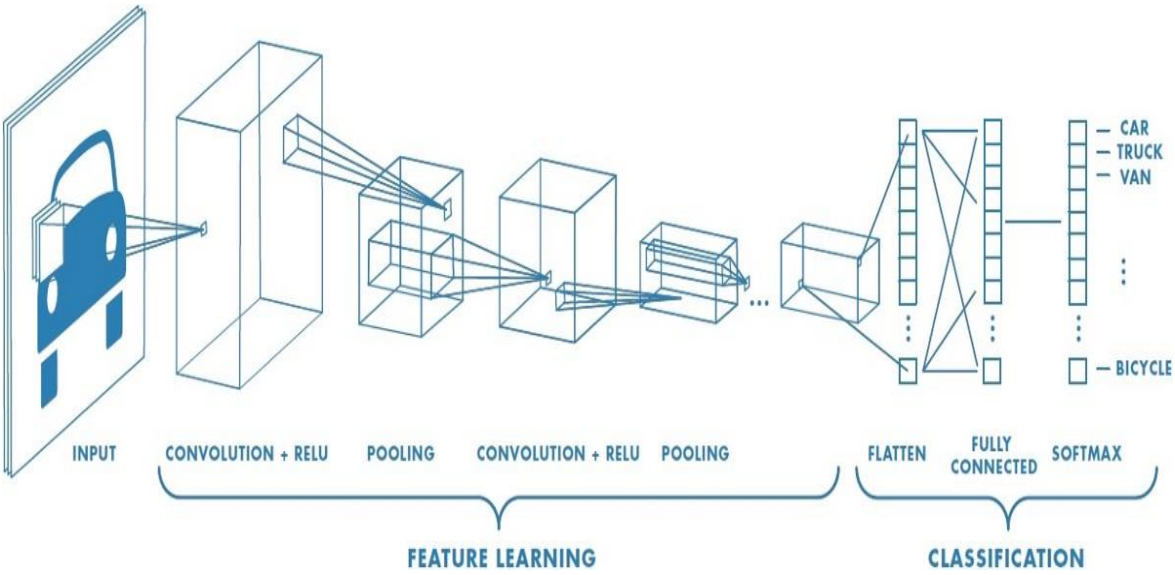

Figure 2.1: Architecture of a Convolutional Neural Network (CNN).

2.2 3D Object Detection from Images

Image based 3D object detection is cheaper and only requires single or pair of camera modules to capture raw RGB image data. It is easier to process the image data as compared to the LiDAR data. The image-based methods can be classified into monocular and stereo. Mono3D [35] performs 3D object detection from a monocular or single image. The network consists of a CNN that extracts convolutional features from the input RGB image and further splits it into two branches viz., context and proposal regions that use region of interest (RoI) pooling and fully connected layers encoded features. The final feature vectors are obtained by combining features from both the branches and are used to classify the object and regress the bounding box parameters such as orientation and dimensions. Deep3DBox [9] first performs 2D object detection and uses 2D bounding box to regress 3D bounding box parameters by leveraging the geometry constraints. SMOKE [34] propose a single stage monocular 3D object detection architecture that pairs each object with a single key-point. Unlike other methods, the network eliminates computing 2D object proposals and directly regresses the 3D object

proposals. The backbone CNN network extracts feature maps that are provided to the key-point classification and 3D box regression branches. 3DOP [33] exploits information from stereo images of contextual features such as object priors, ground plane as well as depth features like point densities to generate object proposals.

2.3 3D Object Detection from LiDAR

The fact that LiDAR data can be represented as 3D point clouds has led to the recent progress in 3D Vision and LiDAR-based 3D Object Detection. Objects in a point cloud are naturally separated in physical space, making it easier to perform segmentation and localization tasks. On the other hand, the pixels of objects in 2D images are near-by each other even though the objects are physically located at a distance from each other. This makes it difficult to estimate accurate depth in images and introduces error for 3D localization.

The state-of-the-art LiDAR-based methods assume that the precise 3D point coordinates are given. PIXOR [20] is a single stage, proposal-free object detector that uses Birds Eye View (BEV) or a 2D representation of the point cloud to train the CNN. The BEV allows to preserve the necessary information in a point cloud in the form of 3 channels (height, intensity and density) that are similar to RGB channels. YOLO-3D [17], extends the YOLO-V2 [32] network to include the yaw angle, the 3D box center coordinates (x,y,z) and the height of the 3D box as a direct regression task. Vote3Deep [21] incorporate a voting scheme and relu [23] non-linearity to process point-cloud data using a CNN. This process is repeated to predict the detection scores. Chen et al., in MV3D [31] propose a multimodal architecture that has a 3D proposal network along with a region proposal network. The network combines features obtained from LiDAR BEV, Front-View (FV) and input image using deep fusion to predict oriented 3D bounding boxes around the objects.

Frustum PointNets [23], perform 2D object detection and lift the 2D regions to 3D, known as frustum proposals. Further, using point cloud coordinates in frustum, they perform 3D instance segmentation around the object and later use a T-NET architecture to estimate 3D bounding box parameters. Ku et al., in AVOD [24] propose a two-stage, multimodal fusion 3D object detection network that generates feature maps from point clouds BEV and monocular RGB images. The feature maps are passed to the region proposal network that regresses the 3D dimensions of the box based on anchor grid. The model achieves state-of-the-art results on the KITTI [13] autonomous driving dataset. MVF [26] propose an end-to-end multi-view feature fusion model that uses dynamic voxelization to fuse information from BEV and perspective view. The model takes a raw LiDAR point cloud as input and embeds each point into a high-dimensional feature space via one fully connected layer. Finally, it uses the extracted point-level context information from multiple views to perform 3D object detection. Point-RCNN [29], propose a two-stage network that directly generates 3D proposals in a bottom-up manner from raw point cloud data, unlike [23,24]. In the second stage it refines the 3D proposals into its canonical coordinates where the network combines the spatial point features and global semantic features from the first stage to refine the confidence.

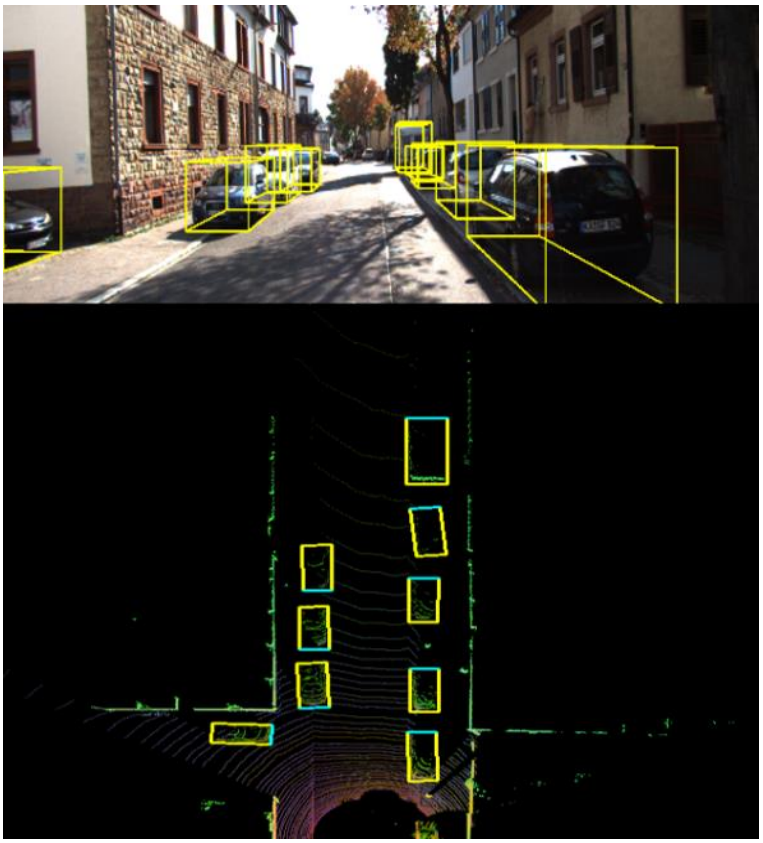

Figure 2.2: A sample output of 3D object detection in Front View and Birds Eye View with 3D bounding box predictions on KITTI [13] dataset.

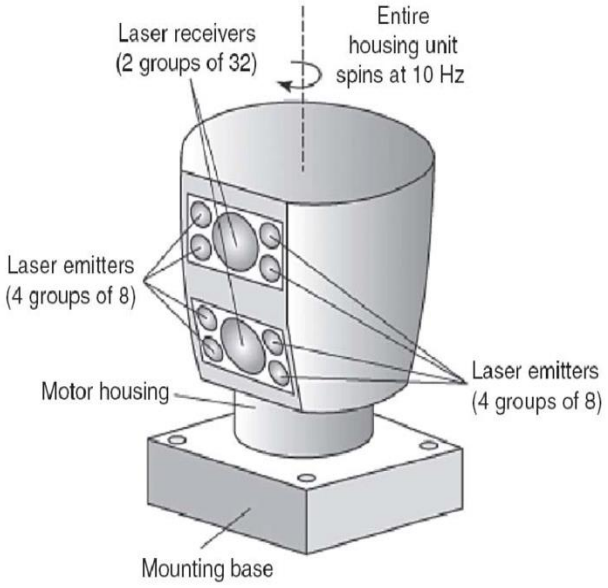

Figure 2.3: Velodyne 64 -Beam LiDAR Scanner Sensor.

2.4 Working Principle of LiDAR

LiDAR scanners emit single laser beam that acquires the horizontal distance between the sensor and the target object. LiDAR's typically operate on the 'time of flight' principle to calculate the distance. The laser beam emitted from the LiDAR gets reflected after hitting an object (Example: Car) and returns back to the receiver. The time between the emitted laser pulse and the returned laser pulse in precisely recorded by the sensor. As the speed of light is constant, the distance between the sensor and object can be measured as follows:

$$Distance\ (d) = \frac{Speed\ of\ Light\ (c) * Time\ of\ Flight\ (ToF)}{2}$$

Knowing the orientation and position of the sensor the x,y,z coordinates of the object can be found. All these values are added to the point cloud data and the process is repeated till the whole landscape has been covered. An additional reflectance value *'i'*, that resembles the return strength of the laser beam is also recorded. The intensity value is used in making the LiDAR BEV-image. All these recorded points together represent the LiDAR point cloud. Figure 2.4 shows the working principle of the LiDAR.

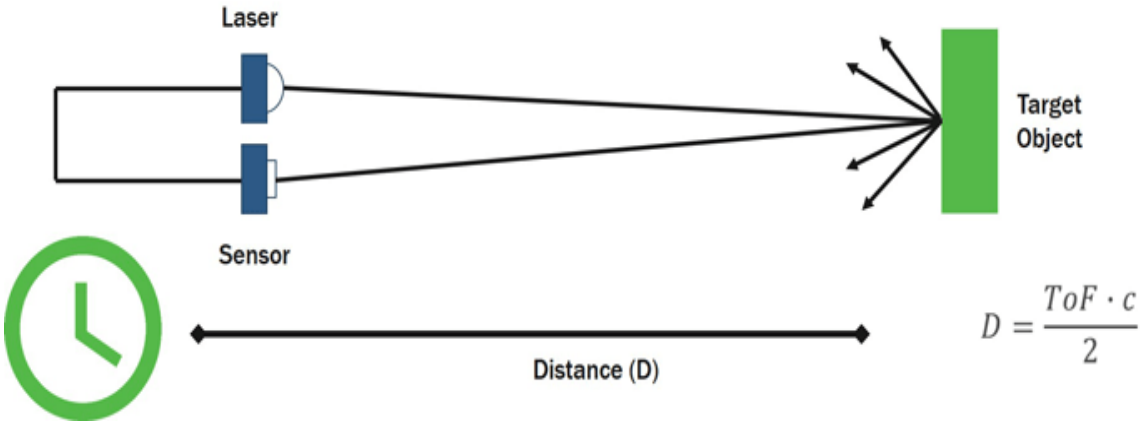

Figure 2.4: Working principle of a LiDAR.

2.5 Depth Estimation

Depth estimation from camera images can be classified into monocular-based and stereo-based. The difference between the two is that the monocular depth estimation uses single RGB image while stereo-based methods use a pair of RGB images. The task of depth estimation includes calculating the disparity 'd' for each pixel in the reference image. Disparity is defined as the difference in the location of the same point when projected under the perspective of two different cameras. The depth is the actual location of the 3D point and can be estimated from the disparity value. For example, consider a pair of rectified stereo images. For the pixel (x,y) in the left image, if the location of the corresponding point is found to be (x-d, y) in the right image, where d is the disparity, then the depth of the pixel can be calculated as $\frac{fB}{d}$ where f is the camera's focal length and B is the baseline or the distance between the two camera centers. Note that disparity is inversely proportional to depth. The depth of every pixel in the reference image results in the formation of a depth map.

The problem of depth estimation can be formulated as a supervised learning task using CNN. In this thesis, we use a state-of-the-art stereo depth estimation network, PSMNet [19]. PSMNet uses spatial pyramid pooling (SPP) network to learn global context information from input images and outputs a combination of left and right feature maps. The features maps are concatenated to obtain a cost volume and are provided to a stack hourglass or encoder decoder CNN module to regularize the feature information. Finally, the aggregated features are regressed to output a disparity map. Figure 2.5 shows a sample depth map image obtained from PSMNet [19].

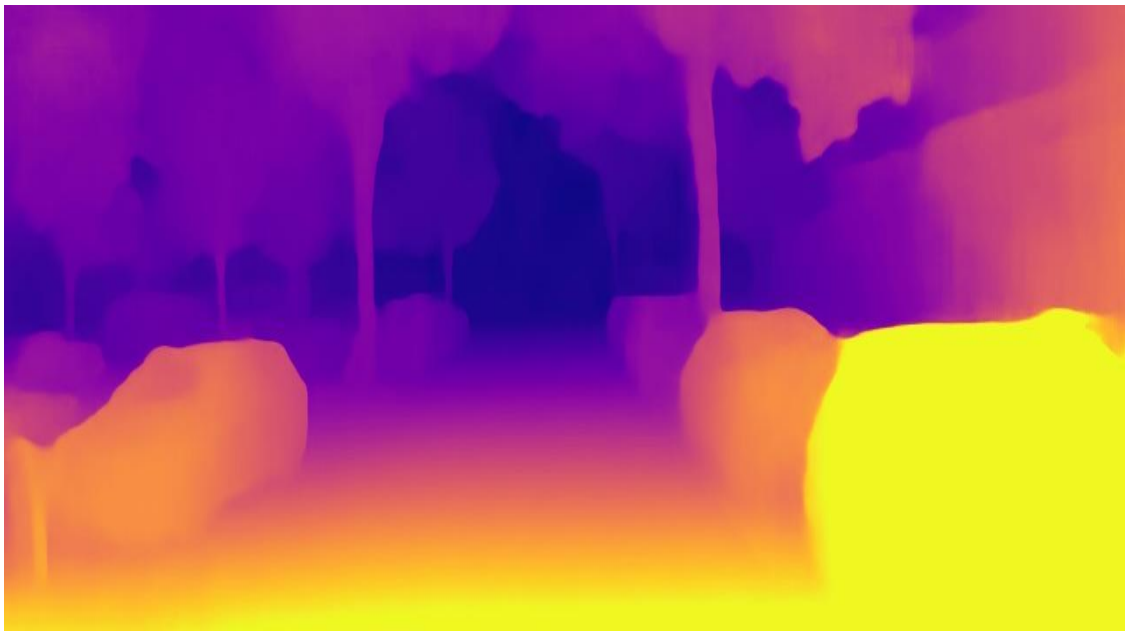

Figure 2.5: Example of depth map from stereo images using PSMNet.

2.6 Pseudo LiDAR

Research suggests that the error in depth calculation for stereo image-based depth estimation grows quadratically with depth. The authors of pseudo-LiDAR [11] claim that it is the representation of the depth data that matters rather that it's quality. To evaluate the claim, they propose a two-stage approach. Firstly, they obtain a depth map from stereo images using state-of-the-art stereo depth estimation methods. Secondly, instead of using the traditional approach of including the depth as additional channel to the image like to RGB-D, the authors of pseudo-LIDAR obtain 3D location of each pixel (u,v) using camera coordinate system.

The dense depth Z (u, v) obtained from stereo images is projected to a 'pseudo-LiDAR' point (x, y, z) in 3D space as given below,

$$\text{(Depth)} \qquad z = Z(u, v),$$

$$\text{(Width)} \qquad x = \frac{(u - cU)^* z}{fU},$$

$$\text{(Height)} \qquad y = \frac{(v - cV)^* z}{fV}$$

where, (cU , cV) is the camera center and fU and fV are the horizontal and vertical focal lengths. The resulting pseudo-LiDAR points in 3D space mimic the LiDAR signal and align fairly with the 3D LiDAR point clouds as shown in figure 2.6. On training the existing 3D object detection CNN algorithms on pseudo-LiDAR point clouds, the authors observe that there is a remarkable improvement in the detection accuracy and that justifies their claim of rightly representing the depth data.

Although pseudo-LiDAR performs significantly well on the task of 3D object detection for IOU value of 0.5, it does not quite match the accuracy of a 3D LiDAR for IOU value of 0.7. The fact that it is generated from a dense depth map instead of a LiDAR, introduces a certain amount of discrepancy for the depth values of points in pseudo-LiDAR that are far-away from the viewpoint. Pseudo-LiDAR++ [12] overcomes this limitation by introducing a stereo depth network and an additional graph-based depth correction algorithm. The depth correction algorithm uses ground-truth depth data from a 4-beam LiDAR as reference points, to correctly match the 3D points in pseudo-LiDAR. Similarly, the remaining points in the pseudo-LiDAR point cloud are iteratively corrected using K-Nearest Neighbors (KNN) algorithm along with KD-Tree search to obtain nearest neighbors in the pseudo-LiDAR point cloud.

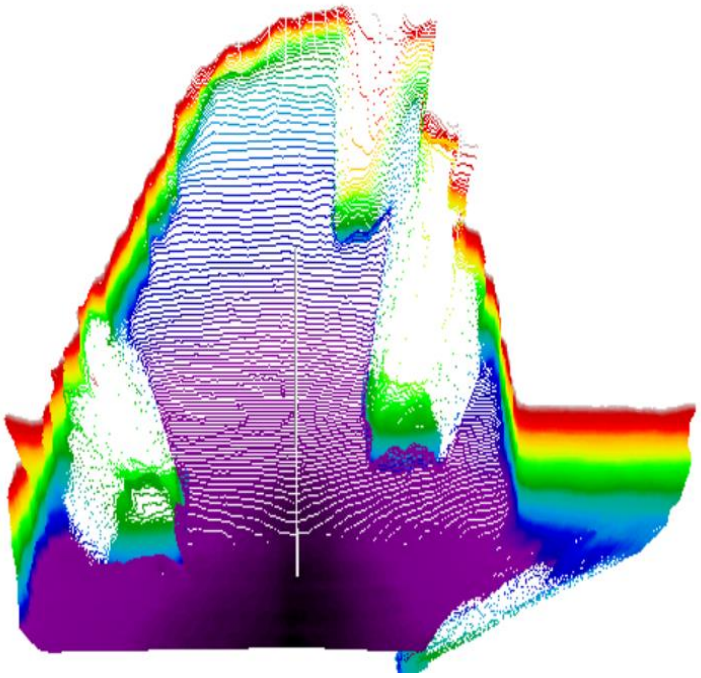

Figure 2.6: An example of dense pseudo-LiDAR point cloud representation generated from dense depth map of stereo images.

2.7 Point Set Registration

Point Set Registration, also known as Point Cloud Registration or scan matching is the process to find an appropriate spatial transformation for two point-clouds by establishing correct feature correspondences between them. Point Set Registration is applied and found to be useful in the domain of autonomous driving, 3D reconstruction, simultaneous localization and mapping (SLAM), virtual and augmented reality, and more. The problem can be outlined as follows:

Consider 2 set of point clouds {P,Q} that have a finite size in a real vector space denoted as R^d. 'P' is the moving point cloud while 'Q' is the static, 'd' is the dimensionality of the vector space and is equal to 3 in case of a 3D point cloud. The goal is to find a

transformation that can be applied to P such that P and Q are aligned and the Euclidean distance between them is minimized. Mathematically, the goal can be formulated as:

$$T^* = argmin_{t \to T}\, dist\{t(P), Q\}$$

where, *T** is the optimal transformation obtained on using a point set registration algorithm and '*T'* denotes set of all possible transformations that the algorithm can search for.

Point set registration methods can be categorized as pairwise and groupwise. The difference between these two methods is that pairwise works only for two point-sets while groupwise can be used for more than two point-sets simultaneously. Furthermore, the type of transformations that can be performed on the point sets are classified into rigid and non-rigid. In a rigid transformation, only the position of points in the point set can be changed in space and does not actually vary the shape and size of the point set. Although, in non-rigid transformation, the structure of the point set can be varied in terms of the shape and size. This can be achieved by using scaling or shearing. Scaling essentially means to increase or decrease the size of the point set.

Chapter 3
Methodology

In this section, we first give the network overview of our proposed network. Then the extraction process of the 2D LiDAR point cloud is discussed. Further, we discuss about the point set registration technique that we incorporate in our proposed method and explain how we implement and achieve depth correction. Finally, we introduce the CNN-based modified CenterNet 3D object detection model and explain it in detail.

3.1 Network Overview

Figure 3.1 shows the diagram for the proposed network. There are two different ways in which the network accomplishes the task of 3D object detection. In the first method, we directly use the single beam point cloud data from 2D LiDAR to train on the 3D object detection CNN network and output the estimated 3D bounding boxes on a 2D image. In the second method, we obtain dense depth map of stereo images using PSMNet [19] and generate pseudo-LiDAR point clouds from the dense depth map. Further, we leverage the single beam point cloud data from 2D LiDAR for depth correction in pseudo-LiDAR using the ACPD [16] algorithm. Finally, the depth corrected point cloud in applied to the 3D object detection CNN to predict 3D bounding box parameters and visualize them on the input image. Figure 3.2 shows a pictorial representation of the two proposed methods in actuality.

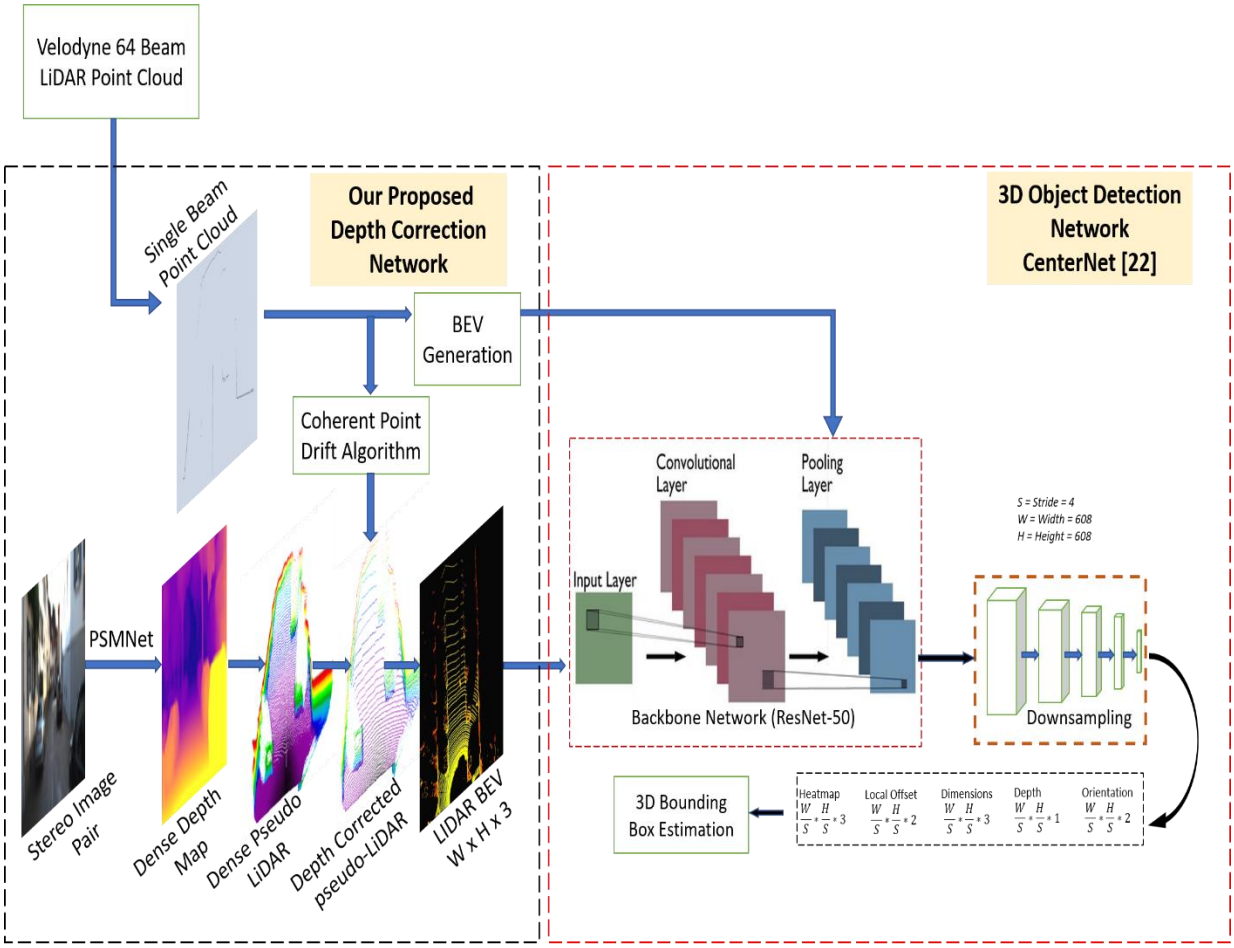

Figure 3.1: The block marked with black lines is our proposed depth correction network. In our first method, we perform 3D object detection using only the single beam point cloud. The corresponding point cloud BEV is provided to the CenterNet [22] based 3D object detection network on the right to predict 3D bounding boxes. Our second method uses the single beam data as reference to correct the disparity in pseudo-LiDAR point clouds by using ACPD [16] algorithm. The depth-corrected point cloud is converted to its corresponding LiDAR BEV and then provided to the CenterNet [22] based 3D object detection network to estimate 3D bounding boxes on input RGB images.

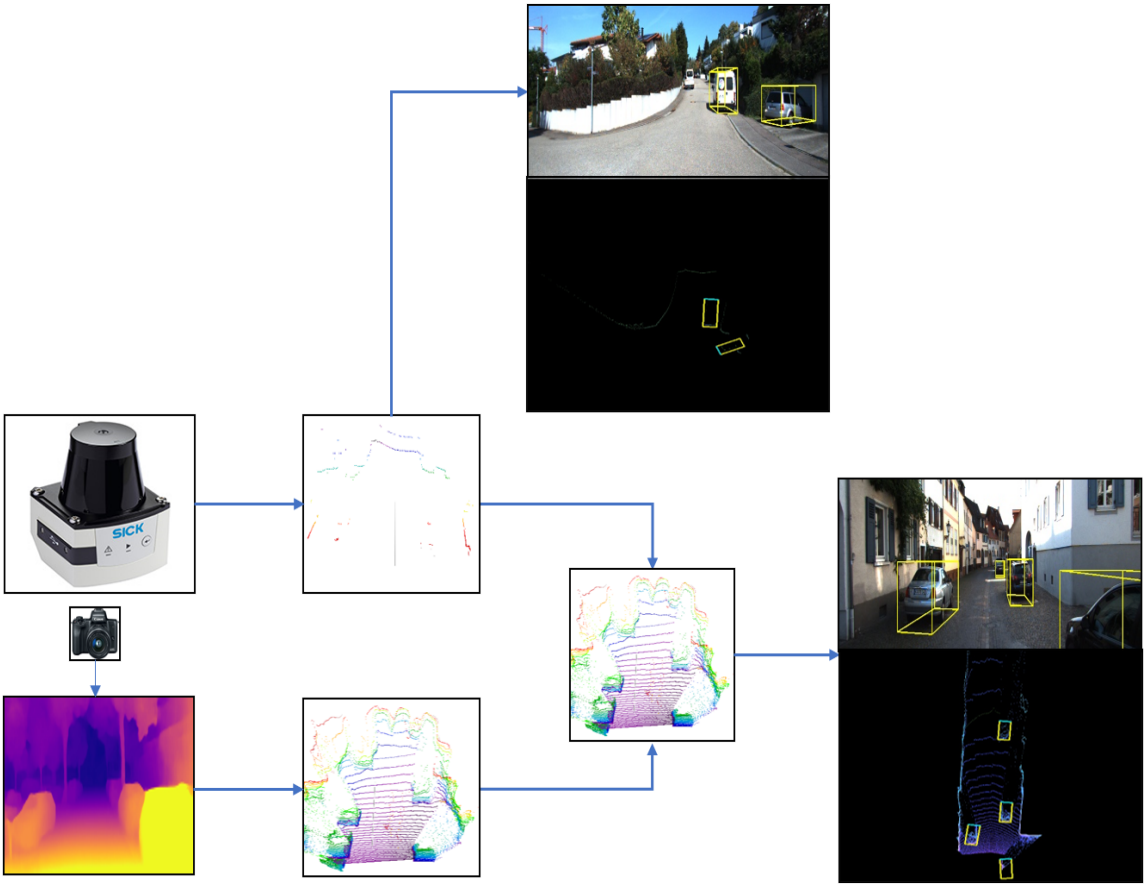

Figure 3.2: Visualization of the proposed methods.

3.2 Extracting Single Beam LiDAR Point Cloud

In this thesis work, the single beam 2D LiDAR data is extracted from a Velodyne-64 beam 3D point cloud. The extracted point cloud is a simulation of the 3D point cloud that can be obtained by using a 2D LiDAR sensor. We follow the algorithm proposed in pseudo-LiDAR++ [12] to extract only the single beam data. The algorithm is fed with a 3D LiDAR point cloud along with a beam number whose points are to be extracted. The point cloud is first converted to a depth map, and based on the specified beam number the particular depth map line is taken out. Further, it is projected back to point cloud format as a single beam 3D point cloud.

Each extracted point cloud consists of about 500-600 points in 3D space. An example of the extracted point cloud is shown in figure 3.3.

For every point $(x_i, y_i, z_i)\ \epsilon\ R^3$ of the point cloud in one scene (in LiDAR coordinate system [x: front, y: left, z: up and (0,0,0) is the location of the LiDAR sensor]), [12] compute the elevation angle θ_i to the LiDAR sensor as $\theta_i = \arg\cos(\frac{\sqrt{x_i^2+y_i^2}}{\sqrt{x_i^2+y_i^2+z_i^2}})$.

The points are ordered by their elevation angles and sliced into separate lines by step of 0.4°. We select LiDAR points whose elevation fall between [-5.6°, -4.4°) to be the 2D LiDAR signal. The elevation angle corresponds to the ninth beam in the 64-beam point cloud. We specifically choose the ninth beam because the LiDAR sensor mounted by KITTI [13] is placed such that the ninth beam passes through most of the objects in the scene.

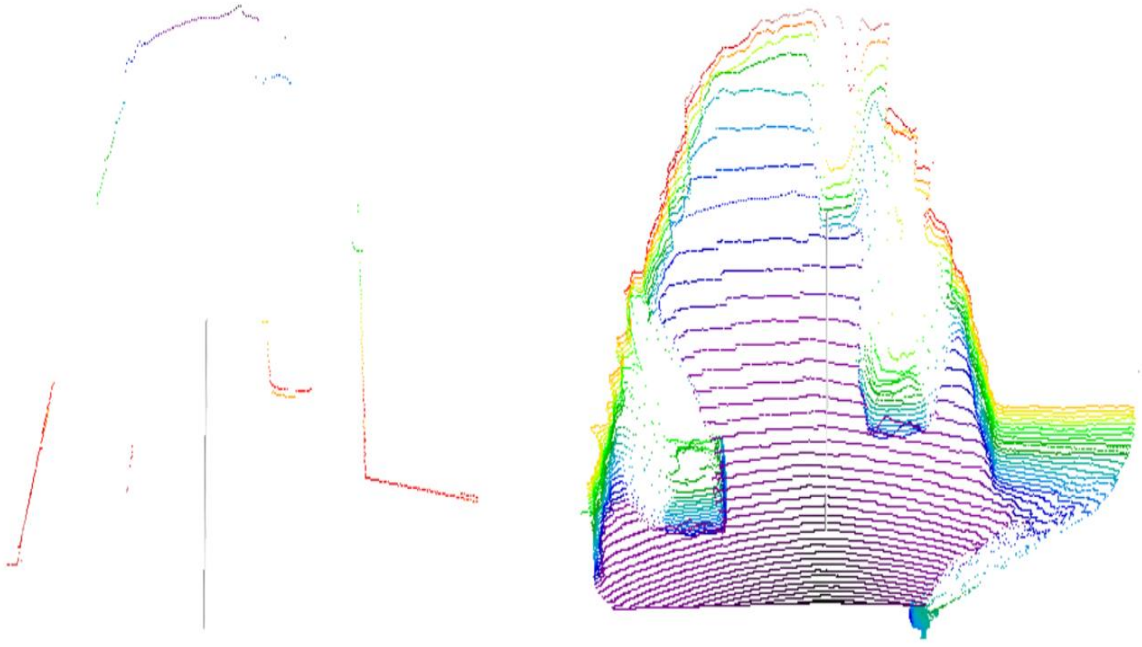

Figure 3.3: Extracted single beam LiDAR point cloud (Left) from Velodyne 64 beam LiDAR point cloud (Right).

Figure 3.4: Extracted single beam, LiDAR to camera projection.

3.3 Accelerated Coherent Point Drift Algorithm

The point set registration modelling methods can be divided into parametric and non-parametric. Examples of parametric models are Iterative Closest Point (ICP) [14], Gaussian Mixture Model (GMM) [15], Coherent Point Drift (CPD) [36], Accelerated Coherent Point Drift (ACPD) [16], Graph matching is an example of non-parametric model.

The ICP is a distance-based method for point set registration and follows a two-step procedure for registration. In the first step, the Euclidean distance between the two point-sets and their correspondences are computed. In the second step, the initial computed distance between the two point-sets is minimized based on the correspondences. The ICP method is very popular, however it requires the point-sets to be coarsely aligned as it is sensitive to initial conditions and is more likely to trap under local minima. The ICP

algorithm is able to achieve accurate registration but has a high computational cost.

The CPD algorithm is a probability-based method in the field of point set registration. In CPD, the registration problem is formulated as a maximum likelihood estimation where one point set is represented using Gaussian Mixture Model (GMM) by GMM centroids and the other point set is coherently fitted to the first by moving the centroids. The likelihood estimation is obtained by using the expectation-maximization algorithm (EM).

Assuming that the correspondences between the points in two point-clouds are not known, any point set registration algorithm arbitrarily assigns a probability to the points based on the proximity values. The CPD algorithm is designed to simultaneously match feature correspondences and estimate the transformation for the moving point set to accomplish the registration task. The correspondence probability is obtained by the E-Step. Further, in the M-Step, the log likelihood is maximized with respect to the transformation parameters. The E-Step and M-Step are repeated until there a good match. As compared to ICP, CPD is more robust towards noise and outliers and performs equally well in terms of accuracy. Although, it suffers from high computational complexity and as the number of points in the point-sets increase the computational complexity also increases.

In order to make fast and accurate point-cloud registration, Lu et al., propose Accelerated Coherent Point Drift algorithm or ACPD [16]. The ACPD has significant improvement in the performance over the original CPD algorithm. ACPD consists of two major changes as extension to the CPD. Firstly, it uses global squared iterative expectation-maximation or gSQUAREM technique to catalyze the convergence process. Secondly, it combines the gSQUAREM technique with dual-tree improved fast gauss transform or DT-IFGT to speed-up the CPD algorithm.

The ACPD algorithm can be summarized as follows:

a) Initialization Step: Assign probability to moving points based on their proximity and calculate variance of distances between all the possible pairs
b) gSQUAREM optimization to compute probability for DT-IFGT:
    1) E-Step - Compute probability of matches for all possible point pairs based on current variance.
    2) M-Step - Compute new transformation that increases the probability and update the probability value based on registration.
    3) Repeat until convergence.
c) Generate final geometric transformation and probability matrix.

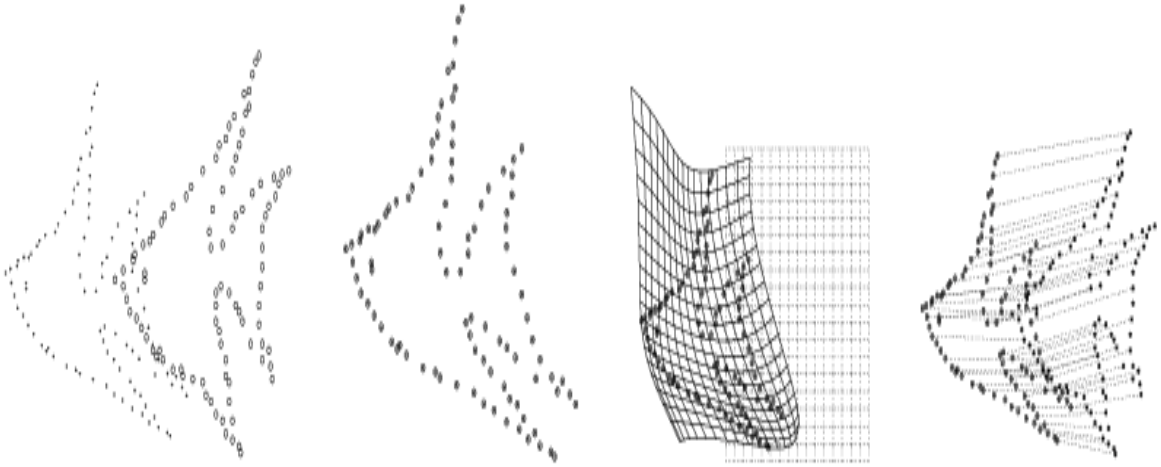

Figure 3.5: In the left most image, two point-clouds are considered. First is the source point cloud and the second one is target point cloud. The right most image shows the probabilistic matching of the two point-sets using Accelerated Coherent Point Drift algorithm

3.4 Depth Correction

The formation of a pseudo-LiDAR point cloud is a result of multiple data processing steps. The data processing introduces small error values for each step. Depth map estimation methods that use CNN, often tend to emphasize nearby objects than the faraway objects. So, the pixel wise depth map obtained by inverting the disparity introduces a bias in the pseudo-LIDAR point cloud. To alleviate the effect of bias in the depth map, it is necessary to correct the gross errors.

To correct the bias, we leverage the ACPD algorithm. The goal is to align the position of certain number of points in pseudo-LIDAR such that the bias is minimized. We consider the pseudo-LiDAR as the moving point set and the extracted single beam/2D LiDAR as the static point set. The ACPD algorithm probabilistically finds a suitable transformation that tries to match the position of the points in pseudo-LiDAR to the corresponding ground-truth points in 2D LiDAR. Although the 2D LiDAR is not fully capable of capturing local object shapes, it provides the exact location of landmark points. The reference of landmark points used along with ACPD algorithm for point set registration, helps in correcting the bias. The transformation of the point cloud is rigid, where only the position of the corresponding points changes but the shape and size remain the same.

3.5 3D Object Detection Network

3.5.1 CNN Network

Zhou et al., propose CenterNet [22], an object detection network that has an anchor-free approach and also does not use non-maximal suppression. The network treats objects as points, in the sense that it estimates the center point of the object or the bounding box and treats that center point as the object itself. The center point of the object is estimated by using a ResNet-50 [25] CNN which is termed as the 'key-point estimation' network. This network also extracts the input image features that are used to regress all the object parameters such as height, width, length, depth, orientation and location.

CenterNet is designed to take an RGB image as input. We modify CenterNet to provide LiDAR birds eye view (BEV) image as the input to the key-point estimation network. This network is trained to generate heatmaps from the LiDAR BEV image input. Peak of the heatmap is considered to be the object center. Along with the heatmap head, the key-point estimator network predicts 4 additional heads. The second head is for dimensions (height, width and length) of the object, third one is for local offset prediction, fourth one is for calculating depth of the object center point and fifth one regresses the heading angle or orientation of the bounding box.

3.5.2 Birds Eye View (BEV) Image Generation

A point cloud BEV image results in faster training due to 2D convolutions and has lower latency as compared to the range view. It also keeps the object size and distance consistent with respect to the range. A 3-channel dense interpolated BEV image is generated by using the point cloud data. The height, intensity and density data from the point cloud is encoded into one BEV image consisting of 3 channels. Table 3.1 and 3.2 show the point cloud boundaries that are considered for BEV image generation. We

detect only those objects that lie within the image plane. The ones lying outside the image plane are ignored, as they are not labeled in the dataset.

Point Cloud Boundaries for BEV – Front side of Vehicle (in meters)	
Minimum X	0
Maximum X	50
Minimum Y	-25
Maximum Y	25
Minimum Z	-2.73
Maximum Z	1.27

Table 3.1: Point Cloud Boundaries for Birds Eye View - Front side of Vehicle.

Point Cloud Boundaries for BEV – Back side of Vehicle (in meters)	
Minimum X	-50
Maximum X	0
Minimum Y	-25
Maximum Y	25
Minimum Z	-2.73
Maximum Z	1.27

Table 3.2: Point Cloud Boundaries for Birds Eye View - Back side of Vehicle.

3.5.3 Training

For an input BEV-image $I \in R^{W * H * 3}$ the key-point network produces a heatmap $\hat{Y} \in [0,1]^{\frac{W}{S} * \frac{H}{S} * C}$ where *S* is the down-sampling factor and *C* is the number of classes. For training the network we use S = 4, and C = 3 (i.e., Car, Person and Cyclist). The discretization error introduced by the output stride is handled using a local offset $\hat{O} \in R^{\frac{W}{S} * \frac{H}{S} * 2}$ for each center point.

Features from the peak of the heatmap are used to regress the size of the bounding box. The dimensions of the 3D box are predicted as $\hat{S} \in R^{\frac{W}{S} * \frac{H}{S} * 3}$. Depth is computed as an additional output channel $D \in [0,1]^{\frac{W}{S} * \frac{H}{S}}$. For predicting the orientation or the heading angle (yaw), the imaginary and real fractions [*sin(yaw), cos(yaw)*] are directly regressed using *L1* loss. It is denoted as $\Theta \in R^{\frac{W}{S} * \frac{H}{S} * 2}$.

All the features from the backbone network are passed to a separate 3 x 3 convolutional layer, ReLU [27] and further to a 1 x 1 convolution. The bounding box location is calculated using the predicted size and offset together.

The input size of the LiDAR BEV-image is set to 608 x 608 x 3. The BEV image is encoded with the height, intensity and density channels that preserve the necessary point cloud information and is used as a 2D-image to train the 3D object detection model.

3.5.4 Inference

At the inference time, a three channel BEV image is provided as an input to the key-point network. The network extracts the peaks in the heatmap for each class independently. A 3 x 3 max-pooling operation is applied on the heatmap to keep only 50 predictions whose

center confidence is larger than 0.2. The eight corner points are predicted as $\hat{A} \epsilon R^{\frac{W}{S}*\frac{H}{S}*8}$. The orientation angle is calculated by

$$\Theta = \arctan\left(\frac{imaginary}{real}\right)$$

A total of 7 target values are obtained to draw a 3D bounding box around the object. The predicted 3D bounding box parameters are used to draw a 3D box on the corresponding input RGB image. Figure 3.6 shows our 3D object detection inference network with the output image.

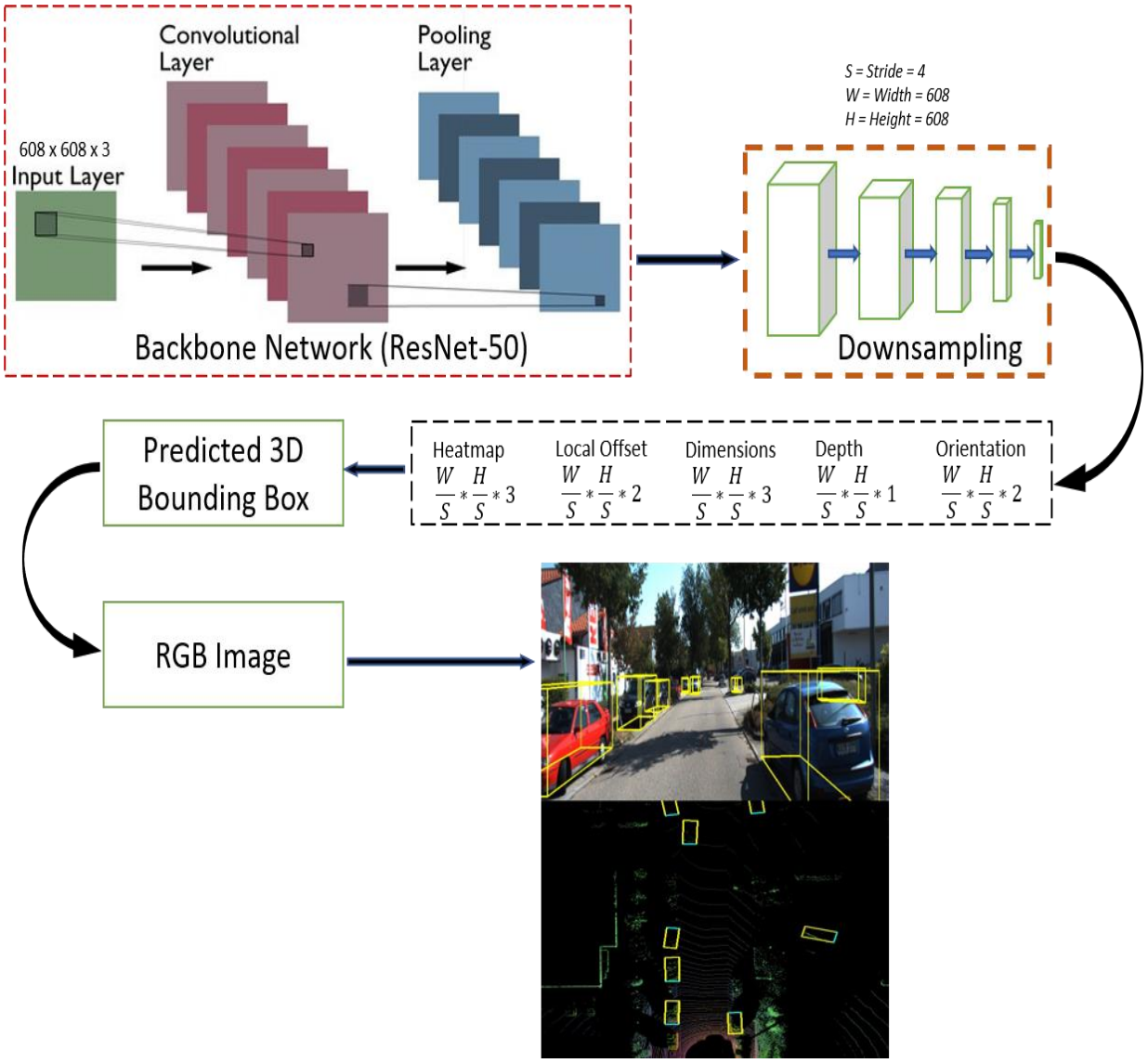

Figure 3.6: The diagram shows the 3D object detection inference phase. The LiDAR BEV image is given as input to the ResNet-50 key-point extraction network. The predicted bounding box parameters are visualized on the corresponding input RGB image.

Chapter 4

Implementation

4.1 Dataset

The KITTI [13] autonomous driving dataset was released in 2012 and has been recorded in the city of Karlsruhe in Germany. In this thesis work, we use the 3D object detection segment of the dataset to perform our experiments. The dataset includes a total of 7481 stereo images along with their corresponding 64 beam LiDAR point clouds, camera calibration matrices and ground-truth labels. The train-test split used for experimentation consists of 3712 images for training and 3769 images validation. Figure 4.1 shows sample images from the dataset.

Figure 4.1: Sample RGB images from KITTI.

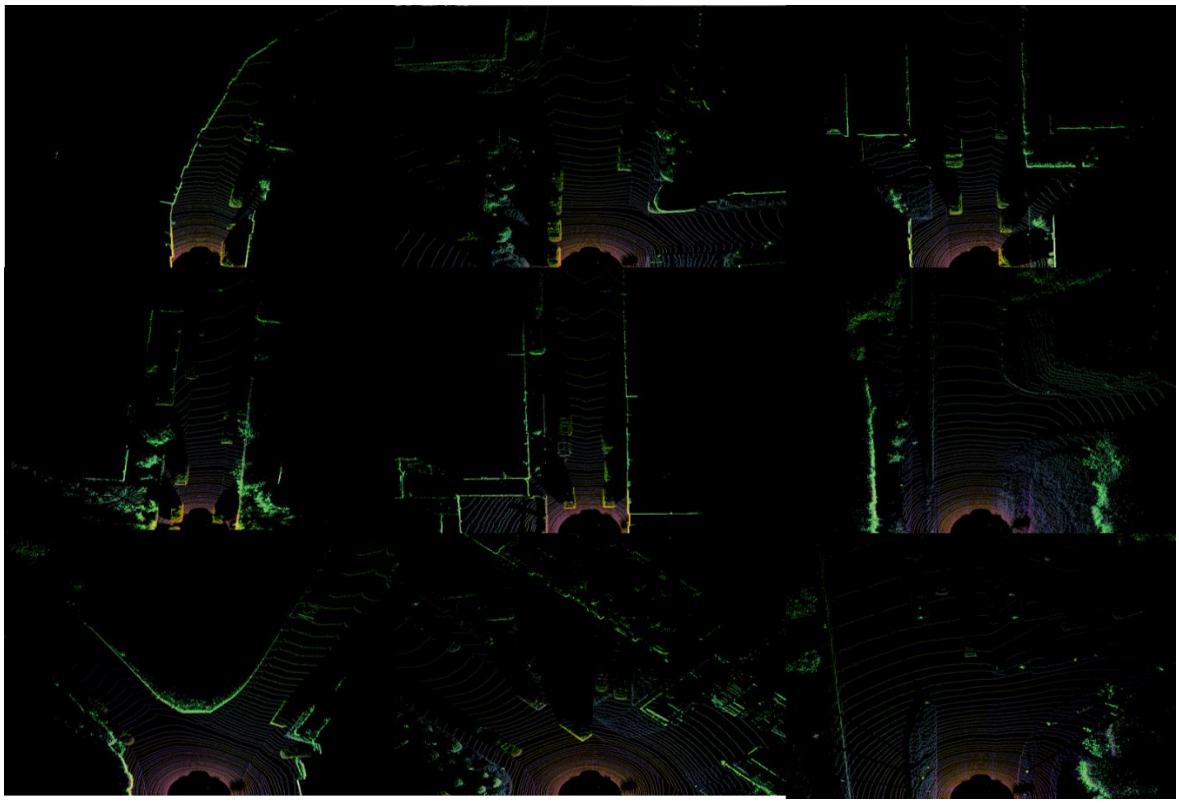

Figure 4.2: Sample Birds Eye View (BEV) images of Velodyne 64 beam LiDAR from KITTI.

4.2 Implementation

4.2.1 Losses

For Center Heatmap, a low-resolution key-point $\acute{p} = \frac{p}{R}$ is computed for each ground truth keypoint $p \, \epsilon \, R^2$. All the ground truth key-points are spread out onto the heatmap $\hat{Y} \, \epsilon \, [0,1]^{\frac{W}{S} * \frac{H}{S} * C}$ using a gaussian kernel $Y_{xyc} = \frac{-\{(x-\acute{p}_x)^2+(y-\acute{p}_y)\}^2}{2\sigma_p^2}$ where $\boldsymbol{\sigma}_p$ is the object size adaptive standard deviation and $\hat{Y}_{xyc} = 1$ corresponds to a detected key-point while $\hat{Y}_{xyc} = 0$ is the background. To assign penalty, the network uses focal

loss. Mathematically, it can be formulated as:

$$L_k = -\frac{1}{\mathrm{N}} \sum_{xyc} \begin{Bmatrix} \left(1 - \hat{\mathrm{Y}}_{xyc}\right)^{\alpha} \log\left(\hat{\mathrm{Y}}_{xyc}\right) \ldots If\ Y_{xyc} = 1 \\ \left(1 - \hat{\mathrm{Y}}_{xyc}\right)^{\beta} \log\left(\hat{\mathrm{Y}}_{xyc}\right)^{\alpha} \ldots Otherwise \\ \log\left(1 - \hat{\mathrm{Y}}_{xyc}\right) \end{Bmatrix}$$

Where, α and β are the hyper-parameters of the focal loss and N is the number of key-point in the image I. During training, the hyper-parameter values are α = 2, β =4.

For the offset and heading angle, we use the L1 loss or Mean Absolute Error (MAE). L1 loss is the sum of absolute differences between the target and predicted labels.

$$L_{offset} = \frac{1}{\mathrm{N}} \left|\hat{\mathrm{O}}_{\acute{\mathrm{p}}} - \left(\frac{p}{R} - \acute{\mathrm{p}}\right)\right|$$

For z coordinate and the 3 dimensions, we use the balanced L1 [28] loss. The balanced L1 loss is derived from the smooth L1 loss. It promotes gradients from accurate samples that facilitates balanced training to accurately perform localization and classification. There are two factors $\alpha\ and\ \beta$ that control the objective function to facilitate more balanced training. The increase in alpha leads to increased gradients for inliers. Gamma is used for tuning upper bound regression errors. The balanced L1 loss can be formulated as:

$$L_b(x) = \begin{Bmatrix} \frac{\alpha}{b} \left(b|x| + 1\right) \ln\left(b\,|x| + 1\right) - \alpha|x| & \ldots if\ |x| < 1 \\ \gamma(x) + C & \ldots Otherwise \end{Bmatrix}$$

4.2.2 Hyper Parameters

The CNN network is implemented using the following hyper parameters.

Training	
Input BEV Image Size	608 x 608 x 3
Learning rate	5e-4
Optimizer	Adam
Epochs	80
Activation	ReLU
Inference	
Input RGB Image Size	1024 x 375 x 3
Input BEV Image Size	608 x 608 x 3

Table 4.1: Input dimensions and hyper parameters used for training and inference. We use Pytorch framework for implementing the network. We train our network on 1 Tesla V100 GPU. The training is done over 80 epochs and takes one day for completion.

4.3 Evaluation Metric

4.3.1 Intersection Over Union (IoU)

Intersection over union, commonly known as IoU is a well-known metric that is used to evaluate the performance of classification, segmentation, object detection methods. The IoU is the ratio of area of intersection over the area of union. In object detection, the IoU is calculated as the area of intersection of the predicted bounding box with the ground-

truth label box divided by the area of union of the two boxes. Figure 4.3 shows the formulation of IoU metric where *B1* is the ground truth box while *B2* is the predicted box. The IoU threshold typically ranges from 0.25 to 0.90. Higher the IoU value, higher is the accuracy of the prediction. The IoU metric is commonly used to calculate mean average precision (mAP) for the predicted labels.

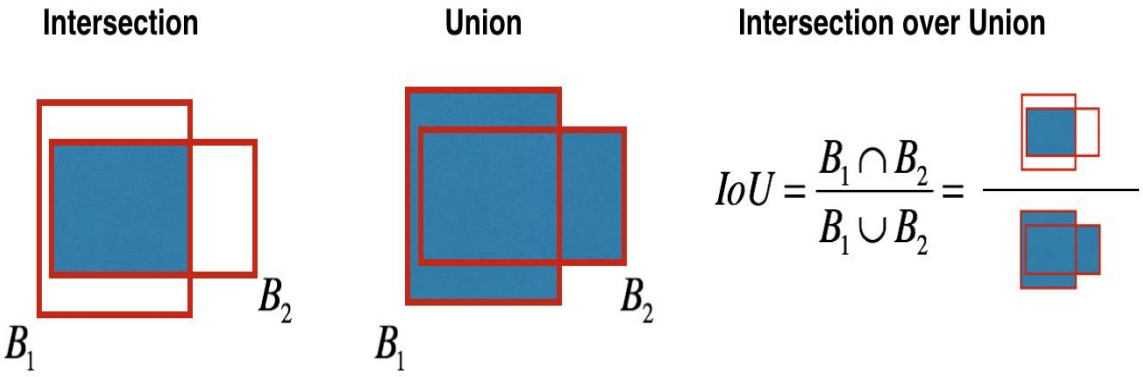

Figure 4.3: Formulation of Intersection over Union (IoU).

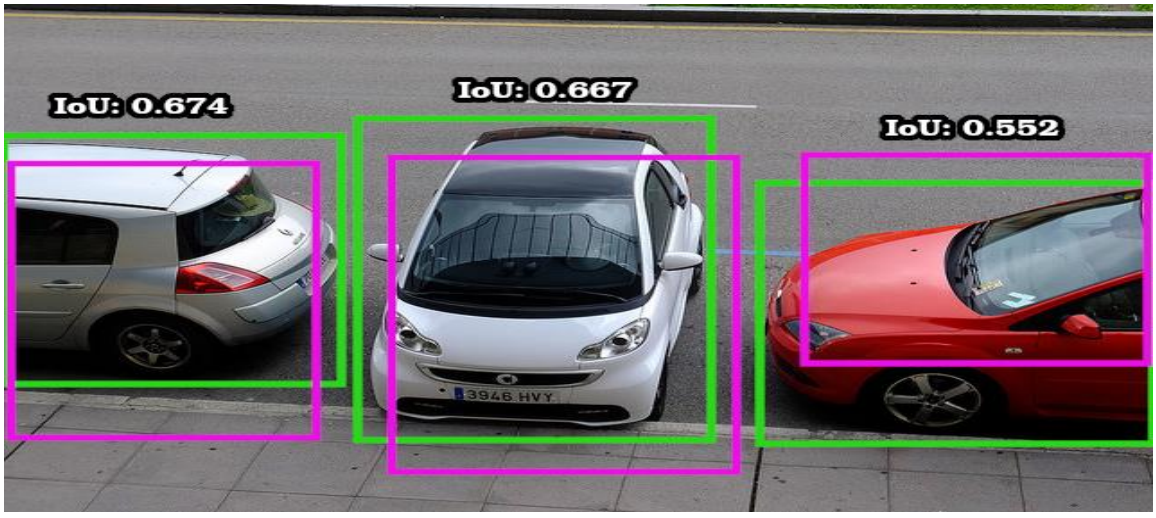

Figure 4.4: Visualization of Intersection over Union (IoU).

4.3.2 Mean Average Precision (mAP)

A large number of object detection methods use average precision (AP) as the metric for evaluating the model's performance. Average precision is the area under the precision recall curve. Precision can be defined as the fraction of valid instances amongst all the retrieved instances. It can be formulated as,

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Recall is defined as the fraction of retrieved instances amongst all valid instances. It can be formulated as,

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Based on the precision and recall values, the average precision and mean average precision can be calculated as,

$$Average\ Precision\ (AP) = \sum_{k=0}^{k=n-1} [Recall(k) - Recall(k+1)] * Precision\ (k)$$

Where, *Recall(n)=0, Precision(n)=1* and *n=number of thresholds.*

$$Mean\ Average\ Precision\ (mAP) = \frac{1}{n} \sum_{k=1}^{k=n} AP_k$$

Where, AP_k *=AP of class k, n = number of classes.*

Chapter 5

Results and Analysis

5.1 Quantitative Results

We evaluate the results of the single beam and depth corrected point cloud on our modified 3D object detection network for the Car and Person classes. Table 5.1 shows the comparison results of our methods with the existing pseudo-LiDAR based methods. Our results show that it is possible to perform 3D object detection using only a 2D LiDAR and achieve fair results. Although, the accuracy of 3D detection is low for 2D LiDAR. The reason for this is that, contextual and depth information obtained from a single beam is limited to the points captured by the 2D LiDAR. A single beam LiDAR captures about 500 points as compared to the 36000 points captured by a 64 beam Velodyne LiDAR.

Our novel depth correction approach that uses point set registration achieves a **4%** and **4.9%** increase in the detection accuracy for the car class on IoU values of 0.5 and 0.7 respectively as compared to pseudo-LiDAR [11] method. For a fair comparison with our proposed depth correction approach, we use the extracted single beam point cloud with the graph-based depth correction algorithm in PL++ [12]. We observe that our depth corrected pseudo-LiDAR achieves comparable performance to PL++ for the car class. We also evaluate our results for the person class, on IoU = 0.5, following the KITTI standards. Although we do not outperform the pseudo-LiDAR ++ [12] in terms of accuracy, our method certainly does better in terms of average inference time.

The inference time for depth correction algorithms (PL++ and Ours) is calculated on CPU while the total time (T) is calculated as a sum of time required for depth correction (T_{dc}) and time required for prediction (T_p) and can be formulated as $T = T_{dc}$ *(Cpu)* + T_p *(Gpu).* For methods which do not have depth correction, the inference time is calculated only on GPU.

Method	Modality	AP 3D (Car)		AP 3D (Person)	Average Inference Time (ms)
		IoU = 0.5	IoU = 0.7	IoU = 0.5	
Pseudo-LiDAR [11]	Stereo	50.09	26.75	8.73	50 (GPU)
Pseudo-LiDAR++ [12]	Stereo + 2D LiDAR	55.72	31.53	10.65	1430 (CPU + GPU)
LiDAR Velodyne	LiDAR 16 Beam	51.34	27.64	8.75	44 (GPU)
Single Beam (Ours)	2D LiDAR	27.70	12.54	2.42	38 (GPU)
Depth Corrected Pseudo-LiDAR (Ours)	Stereo + 2D LiDAR	52.10	28.08	9.14	1100 (CPU + GPU)

Table 5.1: Comparison results of our method with existing pseudo-LiDAR methods and the Velodyne 32-beam LiDAR.

5.2 Qualitative Results

We evaluate qualitative results for both the 2D LiDAR single beam point clouds and the ACPD depth corrected point clouds. The visualization of 3D bounding boxes is done on the validation RGB images from KITTI [13]. Bounding boxes are also visualized on the

bird eye view or BEV image of the single beam and the depth corrected point clouds.Figure 5.1 through 5.4 show the estimated 3D bounding boxes for validation images on the KITTI dataset for our methods and are compared with the ground-truth and the pseudo- LiDAR++ [12] method. On observing the output of 2D LiDAR, we see that most of the cars and people are detected successfully by the network while some of the predictions are false positives. Another observation suggests that few of the estimated bounding boxes and their orientation is not intact with the object shape. This may be because the 2D LiDAR single beam point cloud is capturing only few hundred points as seen on the birds-eye-view image in figure 5.5. The oriented bounding boxes in figure 5.5 show the detected and localized objects from the aerial view.

The third image shows the output for 2D LiDAR corrected pseudo-LiDAR or depth corrected pseudo-LiDAR. It is observed that almost all objects in the image are detected and the 3D boxes have much better alignment with respect to the location and position of the object. A couple of false positive predictions are seen for both the classes. The increase in the accuracy can be accredited to the higher number of points in the depth corrected point cloud as seen in the figure 5.6.

The fourth image shows the 3D bounding box estimation for pseudo-LiDAR++ method. The results obtained for this depth correction method are slightly better than our method. This is because the pseudo-LiDAR++ uses graph-based depth correction algorithm that tries to correct the depth for the whole pseudo-LiDAR point cloud based on the available ground truth points from the single beam point cloud. While, our method only corrects the depth of the corresponding points.

Figure 5.1: 3D Object Detection and Bounding Box Estimation on KITTI validation data trained on modified Center-Net. Top to bottom: 1) Ground truth boxes, 2) 2D LiDAR (Ours), 3) Depth corrected pseudo-LiDAR (Ours), 4) Pseudo-LiDAR ++ [12].

Figure 5.2: 3D Object Detection and Bounding Box Estimation on KITTI validation data trained on modified Center-Net. Top to bottom: 1) Ground truth boxes, 2) 2D LiDAR (Ours), 3) Depth corrected pseudo-LiDAR (Ours), 4) Pseudo-LiDAR ++ [12].

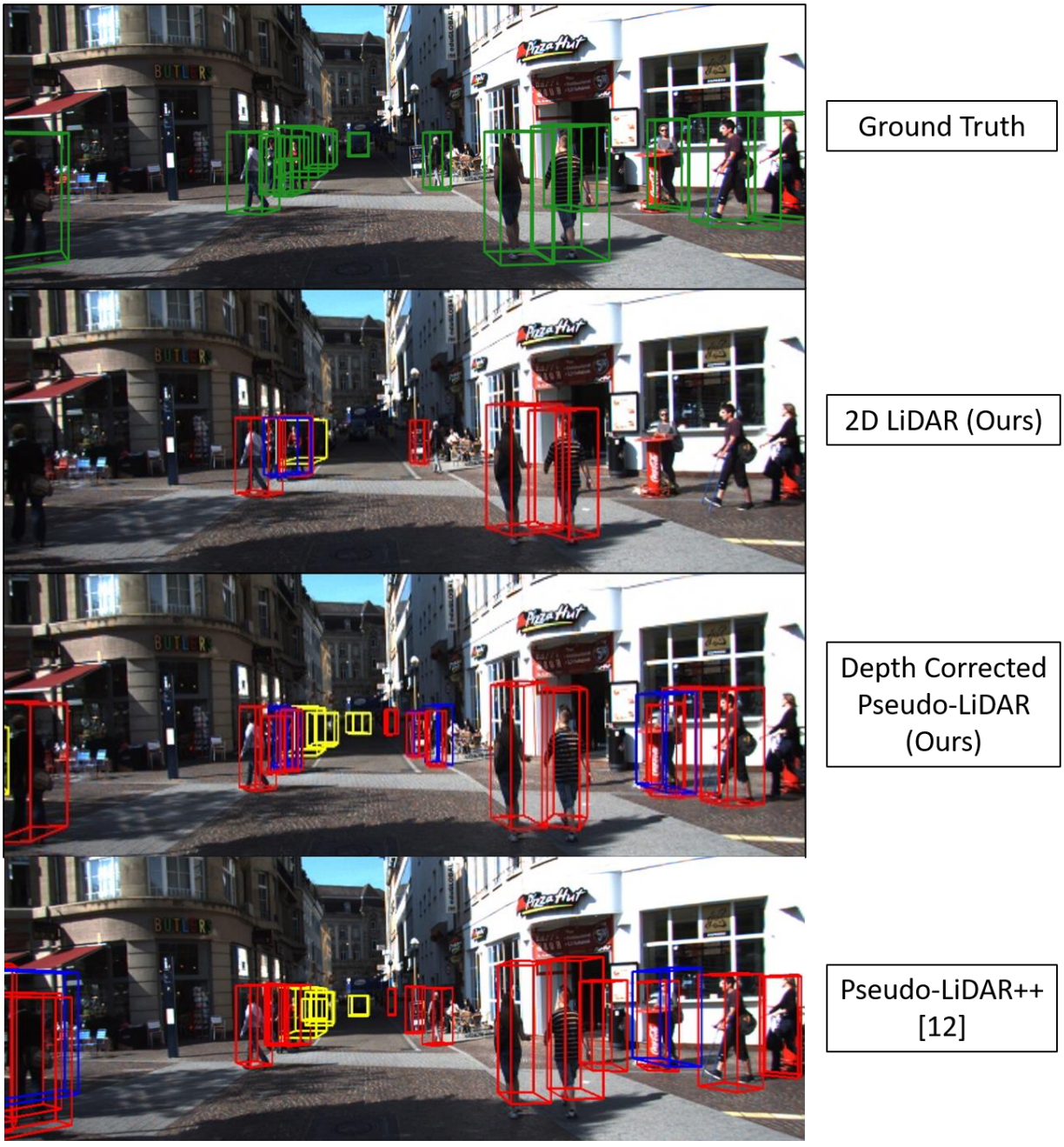

Figure 5.3: 3D Object Detection and Bounding Box Estimation on KITTI validation data trained on modified Center-Net. Top to bottom: 1) Ground truth boxes, 2) 2D LiDAR (Ours), 3) Depth corrected pseudo-LiDAR (Ours), 4) Pseudo-LiDAR ++ [12].

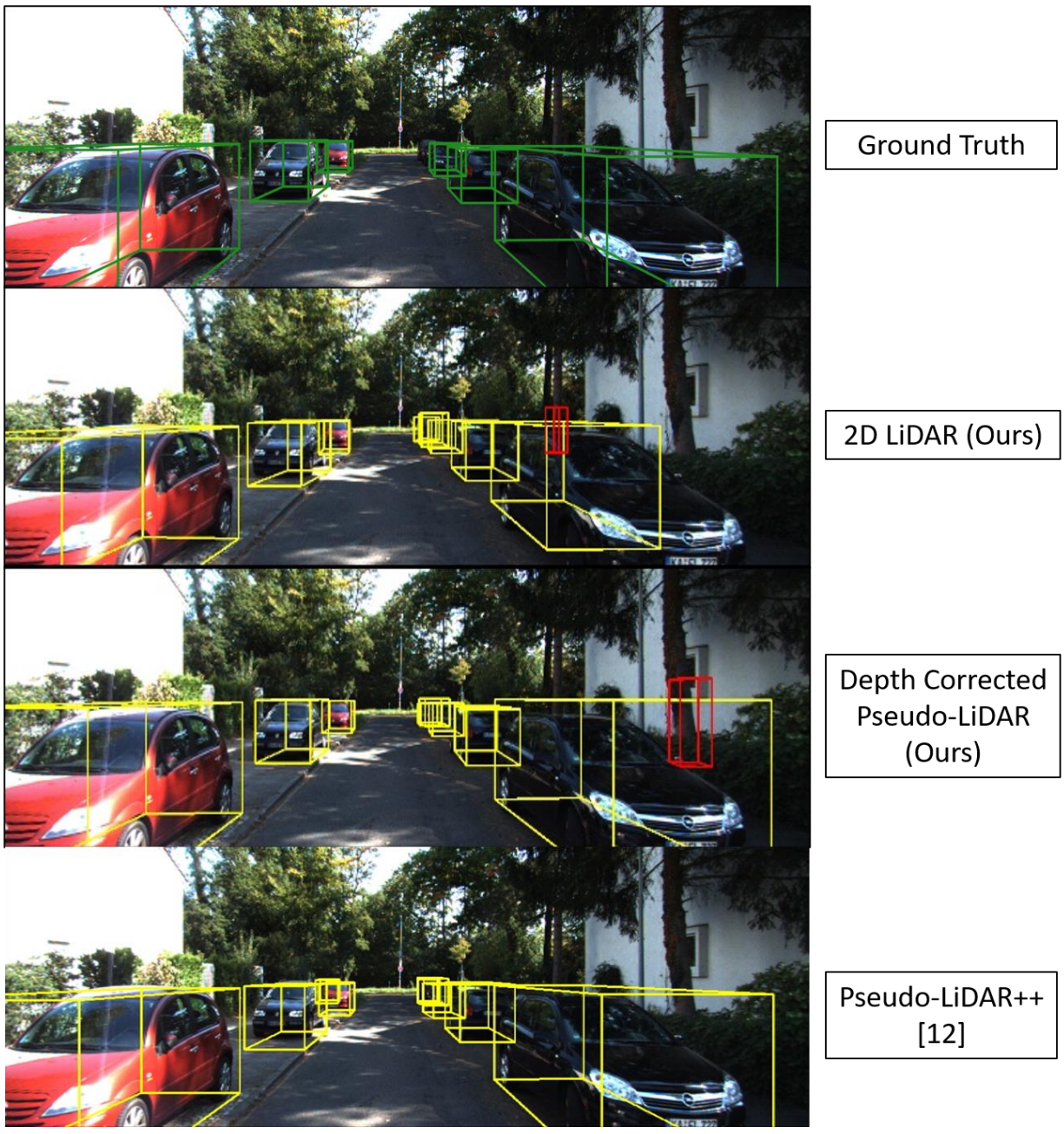

Figure 5.4: 3D Object Detection and Bounding Box Estimation on KITTI validation data trained on modified Center-Net. Top to bottom: 1) Ground truth boxes, 2) 2D LiDAR (Ours), 3) Depth corrected pseudo-LiDAR (Ours), 4) Pseudo-LiDAR ++ [12].

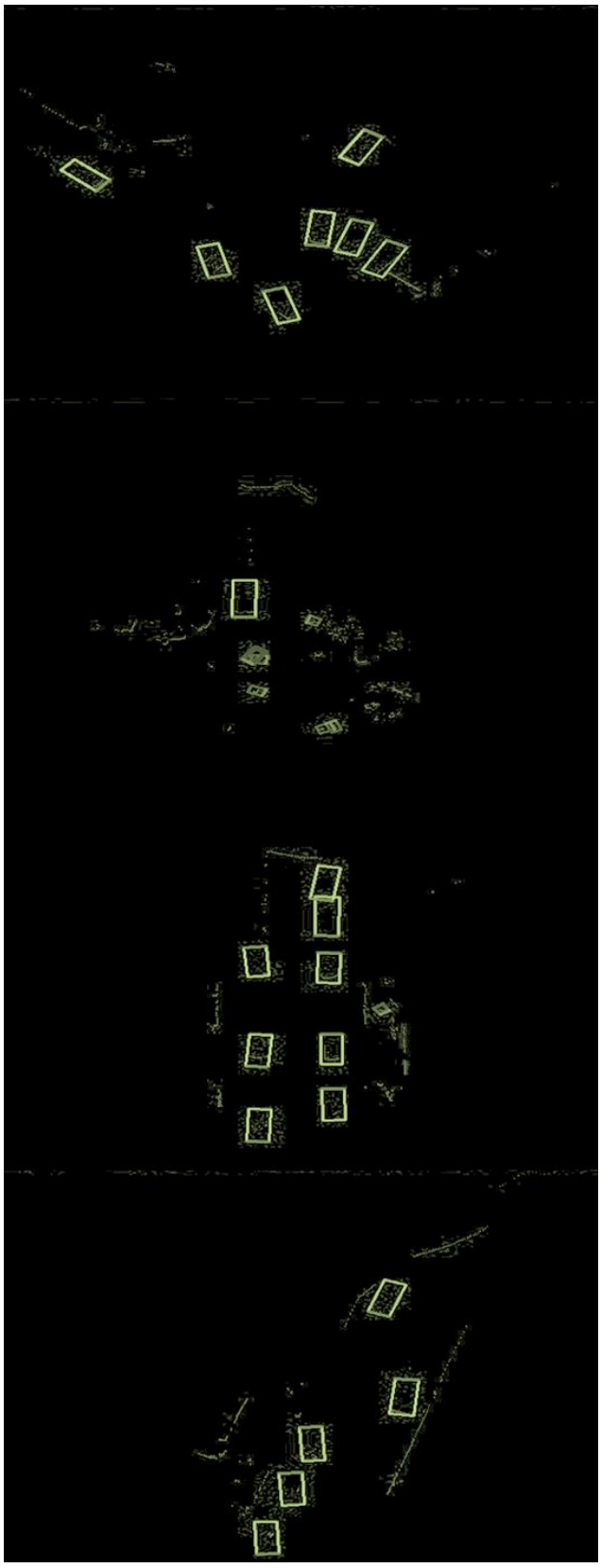

Figure 5.5: BEV Bounding Box Estimation for 2D LiDAR point clouds.

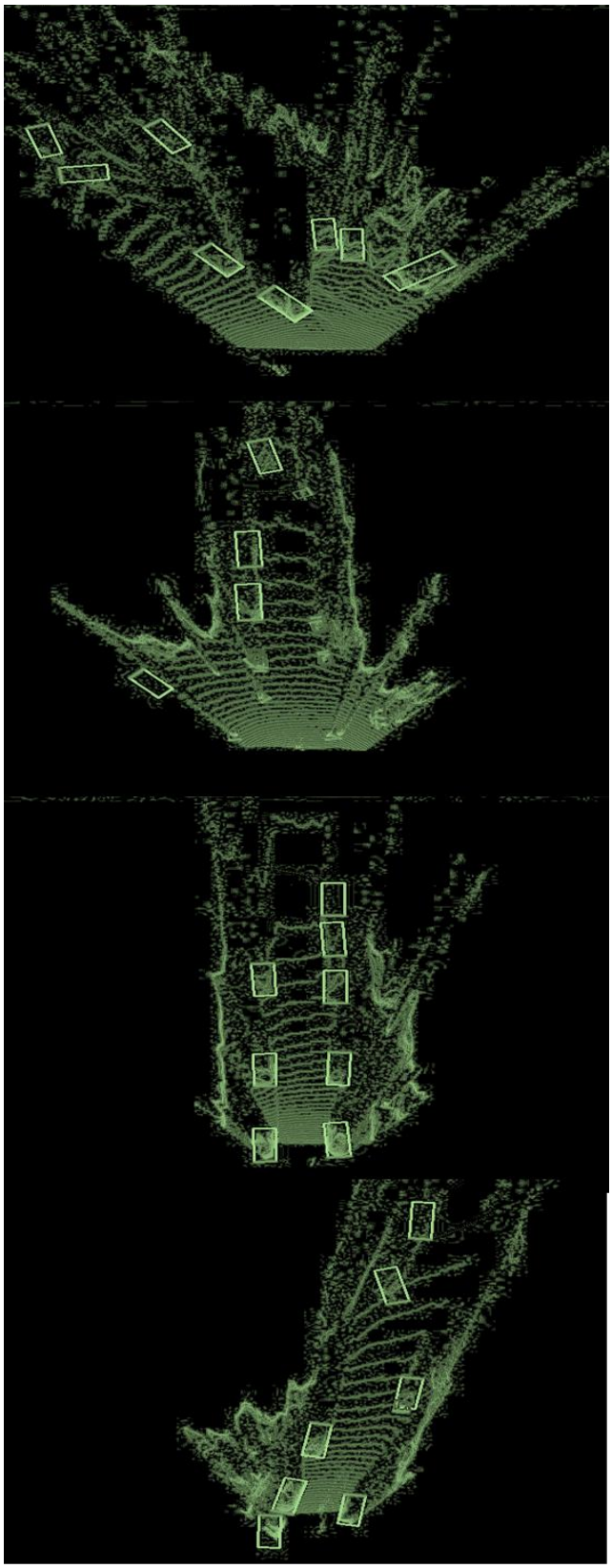

Figure 5.6: BEV Bounding Box Estimation for Depth corrected pseudo-LiDAR point clouds.

Chapter 6

Conclusions

6.1 Conclusion

We leverage 2D LiDAR for 3D object detection and conclude that a single beam is able to perform in comparison to existing monocular image-based methods for the task 3D object detection. The accurate depth information obtained from the 2D LiDAR is further used as a ground truth to correct the disparity in pseudo-LiDAR. We introduced a cost-effective approach for depth correction in pseudo-LiDAR using Accelerated Coherent Point Drift algorithm, a point set registration technique. On comparing with the existing pseudo-LiDAR techniques, our depth corrected point cloud performs better for pseudo-LiDAR but not on par with graph-based depth correction algorithm. However, our proposed methods achieve better inference time. We also achieve a significant reduction in the total cost of the system. Our 3D object detection system costs $2500 approximately while the 64 beam LiDAR system costs $75000. The drastic change in the total cost is achieved majorly because of using the single beam 2D LiDAR and pseudo-LiDAR together instead of multiple beam 3D LiDAR sensor.

6.2 Future Work

As a part of future work,

a) Multi-modal 3D object detection models that implement image and point cloud fusion can be used to train on RGB and LiDAR BEV images together.
b) The time required for depth correction can be reduced by using GPU and CUDA programming.

Experimental Study

1) Angle and beam selection

We recommend the beam number to be chosen from nine to fourteenth beam. These beams are in the middle and are most likely to hit all the objects in the scene as can be seen in the figure below. The computed elevation angle of 0.4 degrees is close to the one provided by KITTI [13] dataset. Based on the suggested beam numbers, the angle to be chosen should be in between -4.4 to -10 degrees. We recommend to mount the LiDAR sensors at a height of 1.3 meters from the ground level.

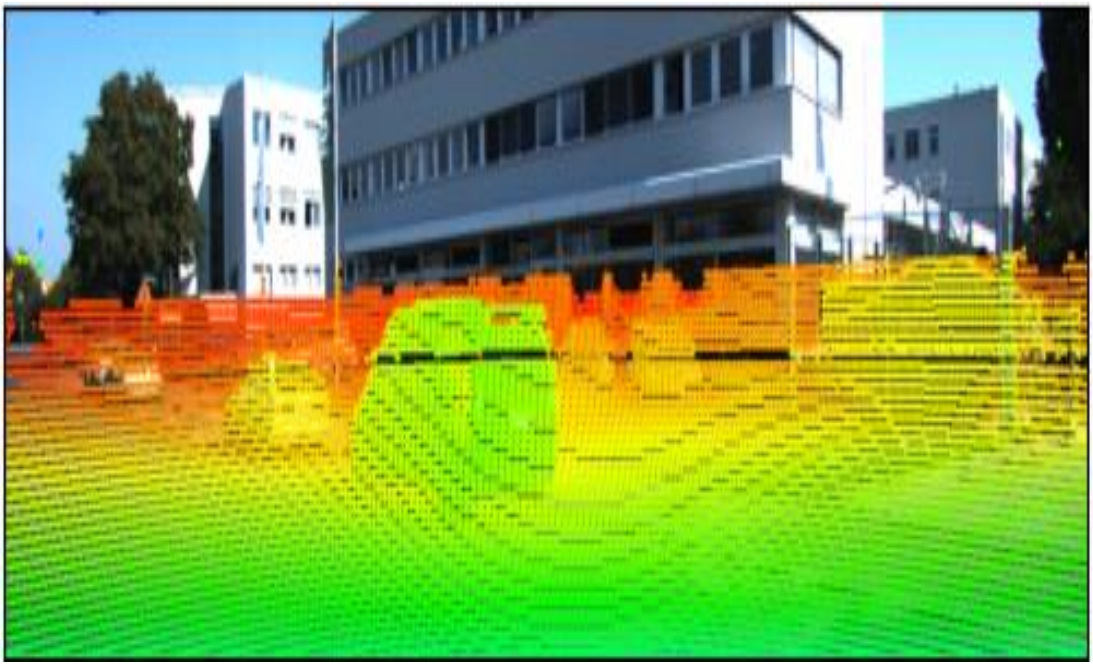

Figure 6. 64-beam LiDAR to camera projection.

2) ICP vs ACPD

We observe that the ACPD algorithm is more accurate that the ICP algorithm. On analyzing the time complexity of both the algorithms we found that the ICP algorithm takes $O(M \cdot N)$ time, where M and N are the number of points in the source and the target

point clouds respectively. While, the time complexity of ACPD algorithm is $O(M + N)$. This shows that the ACPD algorithm is more computationally efficient towards solving the registration problem in 3D point clouds than the ICP algorithm. The high accuracy and efficient registration can be accredited to the gSQUAREM optimization and Dual-Tree Fast Gauss Transform techniques used by the ACPD algorithm. In ICP, computing distances between the two point-clouds requires long time.

Bibliography

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in CVPR, pp. 580–587, 2014.

[2] R. Girshick, “Fast R-CNN,” in ICCV, 2015.

[3] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in NIPS, pp. 91–99, 2015.

[4] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in CVPR, pp. 3431–3440, 2015.

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,” TPAMI, 2018.

[6] Zhu, J.Y., Zhang, R., Pathak, D., Darrell, T., Efros, A.A., Wang, O., Shechtman, E.: Toward multimodal image-to-image translation. In: NIPS (2017).

[7] Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: ICCV (2017).

[8] J. Schlemper, J. Caballero, J. V. Hajnal, A. N. Price and D. Rueckert, "A Deep Cascade of Convolutional Neural Networks for Dynamic MR Image Reconstruction," in *IEEE Transactions on Medical Imaging*, vol. 37, no. 2, pp. 491-503, Feb. 2018.

[9] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Koseck ˇ a. 3d bounding box estimation using deep learning and geometry. In Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, pages 5632–5640. IEEE, 2017.

[10] Buyu Li, Wanli Ouyang, Lu Sheng, Xingyu Zeng, and Xiaogang Wang. Gs3d: An efficient 3d object detection framework for autonomous driving, 2019.

[11] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q.

Weinberger. Pseudo-LiDAR from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In CVPR, 2019.

[12] Yurong You, Yan Wang, Wei-Lun Chao, Divyansh Garg, Geoff Pleiss, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving. In ICLR, 2020.

[13] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

[14] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," Proceedings Third International Conference on 3-D Digital Imaging and Modeling, Quebec City, QC, Canada, 2001, pp. 145-152, doi: 10.1109/IM.2001.924423.

[15] B. Jian and B. C. Vemuri, "Robust Point Set Registration Using Gaussian Mixture Models," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1633-1645, Aug. 2011, doi: 10.1109/TPAMI.2010.223.

[16] M. Lu, J. Zhao, Y. Guo and Y. Ma, "Accelerated Coherent Point Drift for Automatic Three-Dimensional Point Cloud Registration," in *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 2, pp. 162-166, Feb. 2016, doi: 10.1109/LGRS.2015.2504268.

[17] W. Ali, S. Abdelkarim, M. Zahran, M. Zidan, and A. El Sallab, ‘‘YOLO3D: End-to-end real-time 3D oriented object bounding box detection from LiDAR point cloud,’’ Aug. 2018.

[18] Bochkovskiy, Alexey & Wang, Chien-Yao & Liao, Hong-yuan. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection.

[19] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In CVPR, 2018.

[20] B. Yang, W. Luo, and R. Urtasun, ‘‘PIXOR: Real-time 3D object detection from point clouds’’, In Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[21] Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In ICRA, 2017.

[22] Zhou, X., Wang, D., & Krähenbühl, P. (2019). Objects as Points. *ArXiv, abs/1904.07850*.

[23] C. R. Qi, W. Liu, C. Wu, H. Su and L. J. Guibas, "Frustum PointNets for 3D Object Detection from RGB-D Data," 2018 IEEE/CVF Conference on Computer *Vision and Pattern Recognition*, 2018, pp. 918-927, doi: 10.1109/CVPR.2018.00102.

[24] J. Ku, M. Mozifian, J. Lee, A. Harakeh and S. L. Waslander, "Joint 3D Proposal Generation and Object Detection from View Aggregation," 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 1-8, doi: 10.1109/IROS.2018.8594049.

[25] He, Kaiming & Zhang, Xiangyu & Ren, Shaoqing & Sun, Jian. (2016). Deep Residual Learning for Image Recognition. 770-778. 10.1109/CVPR.2016.90.

[26] Zhou, Y., Sun, P., Zhang, Y., Anguelov, D., Gao, J., Ouyang, T., Guo, J., Ngiam, J. & Vasudevan, V (2020). End-to-End Multi-View Fusion for 3D Object Detection in LiDAR Point Clouds. Proceedings of the Conference on Robot Learning, in PMLR 100:923-932

[27] Agarap, A. F. (2018). Deep Learning using Rectified Linear Units

(ReLU). CoRR, abs/1803.08375. Retrieved from http://dblp.uni-trier.de/db/journals/corr/corr1803.

[28] Pang, J., Chen, K., Shi, J., Feng, H., Ouyang, W., & Lin, D. (2019). Libra R-CNN: Towards Balanced Learning for Object Detection. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 821-830.

[29] Shi, Shaoshuai & Wang, Xiaogang & Li, Hongsheng. (2019). PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud. 770-779. 10.1109/CVPR.2019.00086.

[30] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. Proc. Computer Vision and Pattern Recognition (CVPR), IEEE, 2017.

[31] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In IEEE CVPR, 2017.

[32] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6517-6525, doi: 10.1109/CVPR.2017.690.

[33] X. Chen, K. Kundu, Y. Zhu, A. Berneshawi, H. Ma, S. Fidler, and R. Urtasun. 3d object proposals for accurate object class detection. In NIPS, 2015

[34] Liu, Zechen & Wu, Zizhang & Tóth, Roland. (2020). SMOKE: Single-Stage Monocular 3D Object Detection via Keypoint Estimation. 4289-4298. 10.1109/CVPRW50498.2020.00506.

[35] Chen, X.; Kundu, K.; Zhang, Z.; Ma, H.; Fidler, S.; and Urtasun, R. 2016. Monocular 3d object detection for autonomous driving. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2147–2156.

[36] A. Myronenko and X. Song, "Point Set Registration: Coherent Point Drift," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 12, pp. 2262-2275, Dec. 2010, doi: 10.1109/TPAMI.2010.46

[37] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2446–2454, 2020.